In the Specification

Please replace the paragraph beginning at page 2, lines 28 with the following amended paragraph.

The business logic is the only function that is unique to the application. In the N-Tier enterprise model, the business logic is programmed as objects. One common programming language used for enterprise wide applications is a programming language capable of being interpreted by a network browser. One such programming language is often referred to as ~~called~~ ~~JAVA~~JAVA® (a registered trademark of Sun Microsystems). While any programming language could be used, in the description of the preferred embodiment the programming language ~~JAVA~~JAVA® will be referred to. Objects in ~~JAVA~~JAVA® that execute the business logic are often called "beans." In the description of the preferred embodiment, the objects will be referred to as ~~Java Beans~~ JAVAbeans™. However, it should be appreciated that objects created in other languages (also known as software components) might be used and could be tested in the same manner as described herein. Also, it is not strictly necessary that the business logic be implemented in an object-oriented language. The term "object" will be used in conjunction with the preferred embodiment. Also, the term ~~Enterprise Java Bean~~ Enterprise JAVAbean™, or EJB, will be used in conjunction with the preferred embodiment. However, it should be understood that the invention is not limited to testing enterprises that are programmed with an object-oriented language.

Please replace the paragraph beginning at page 7, line 17 with the following amended paragraph.

To enable ASP to conduct testing on an EJB, it must have access to the higher level EJBs from which that EJB has inherited attributes. To provide this access, the business interface 324 downloads to the developer a packer 332. Packer 332 is a program resident on the developer's computer that can find on the computer memory certain EJBs and send them to ASP 320. Packer 332 knows which EJBs to obtain because each EJB includes a portion that describes the

dependencies of that EJB. In ~~JAVA~~JAVA®, an EJB is represented as a .jar (~~JAVA~~JAVA® Archive) file. The .jar file contains the executable code of the EJB in addition to a description of the dependencies of the file. In this way, business interface 324 can figure out all the EJBs that are needed to test a specific EJB. These EJBs could be uploaded to ASP 320 for the test. While a .jar file is described in relation to EJBs, it should be understood that any archive relating to the software component being employed could also be used.

A2

Please replace the paragraph beginning at page 8, line 4 with the following amended paragraph.

It is also possible that the test data could be obtained in an automated or semiautomatic fashion. Each ~~Java bean~~ JAVAbean™ contains a public interface that specifies the methods and properties of that bean. From the interface, it is possible to know which properties must be provided and measured to test a particular method. Thus, a table could be constructed for each method, listing the inputs and the outputs encountered in executing that particular method. A data set for testing the method would have to have data values for each entry in that table.

A3

Please replace the paragraph beginning at page 11, line ~~17~~ 21 with the following amended paragraph.

Further, certain types of tests were described above. The list is intended to be illustrative rather than limiting. ASP 320 might provide other types of tests. It might provide a compliance test—, for example, to verify that each EJB is in compliance with the specifications for objects written in ~~JAVA~~JAVA® or another appropriate language. Alternatively, ASP 320 might test to determine whether a particular EJB uses any features that are not supported by a particular application server.

A4